

Express Mailing Label No.: ER032358151US

PATENT APPLICATION

Docket No.: 1200.2.81

IBM Docket No.: SJO920030020US1

UNITED STATES PATENT APPLICATION

of

Richard Victor Kisley

for

**APPARATUS METHOD AND SYSTEM FOR ASYNCHRONOUS
REPLICATION OF A HIERARCHICALLY-INDEXED DATA STORE**

APPARATUS METHOD AND SYSTEM FOR ASYNCHRONOUS REPLICATION OF A HIERARCHICALLY-INDEXED DATA STORE

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The invention relates to computer systems that increase information availability by maintaining duplicate copies. Specifically, the invention relates to devices, methods, and systems for replicating information from a hierarchically-indexed data store to a backup data store.

2. The Relevant Art

Data processing systems often maintain duplicate storage volumes or other data stores. To create and maintain duplicate data stores, data from a source volume is replicated onto a target volume. Data replication techniques generally include backup and archival storage techniques, data mirroring techniques, and the like.

Some existing data replication systems replicate data in a synchronous, atomic manner in order to maintain coherency between the source and target volumes. Synchronous replication techniques often halt further data processing until an acknowledgment has been received that a write operation performed on the source volume has also been performed on the target volume. Consequently, synchronous replication techniques may significantly degrade data replication performance, particularly when replication takes place over a high-latency communication line.

Multiple techniques have been devised to minimize performance degradation while achieving an acceptable level of coherency between source and target volumes. One technique is periodic full-copy replication, where an entire volume is replicated to a target

1 volume in order to provide an exact copy of a source volume image at a specific time.
2 Disadvantages of periodic full-copy replication include inefficient usage of resources to copy
3 data that has not changed since a previous replication and the necessity to ensure that the
4 source volume is not changed during replication.

5 Another technique for efficient data replication is incremental replication, where the
6 source is replicated on the target as a baseline volume, and thereafter only changes to the
7 source are replicated to the target on a periodic basis. One disadvantage of incremental
8 replication is the necessity to ensure that the source volume is not changed during
9 identification and replication of changed data.

10 Point in time replication techniques, such as snapshot operations, may save the
11 system state information as well as changed data. Such techniques have been used to
12 alleviate the need to suspend processing during incremental replication. Some data
13 replication systems that require continual access to replicated data use snapshot techniques to
14 implement data mirroring. Currently available snapshot replication techniques facilitate
15 restoration of a source volume to a previous desired state and restoration of a particular
16 image on the source volume.

17 One existing method for conducting snapshot-based replication involves saving a
18 periodic snapshot of a source volume within an incremental system log and replicating both
19 the log (including the incrementally changed data) to a target volume, where the snapshot
20 includes all data modifications made to the source volume since a previous snapshot. Using
21 an incremental log simplifies the process of replicating changes to a target system. Since the
22 replication is based on specific snapshots, replication of the source image may occur as
23 asynchronous to the actual write operations.

24 Figure 1 is a block diagram illustrating a typical prior art log-structured mirrored data
25 store 100 and associated snapshots. The depicted log-structured mirrored data store 100
26 includes a source volume 100a and a target volume 100b. As depicted, the source volume
27

1 100a includes one or more snapshot instances 120 and file writes 122, and the target volume
2 100b includes target snapshot instances 130 and replicated files communicated from the
3 source volume 100a.

4 Typically, the file write entries 122 correspond to data written to sequential logical
5 data blocks on a storage device corresponding to the source volume 100a. Each snapshot
6 instance 120 may contain a copy of a current block map for the source volume 100a at the
7 time that the snapshot instance 120 is taken. The block map is a master index linking file
8 names and attributes with the associated data blocks that constitute the files.

9 The target file writes 132 are copies of the source file writes 122, written
10 asynchronously to the target volume 100b after creation of the snapshot instance 120. The
11 target snapshot instances 130 may contain copies of the current block map for the target
12 volume 100b written to the target volume log directly following the target file writes 132.
13 The target snapshot instances 130 may be identical to the corresponding source snapshot
14 instances 120.

15 While the depicted approach creates a complete and consistent image of a source
16 volume by replicating only the changed data, the depicted approach functions only on
17 systems that use log-structured file systems.

18 Figure 2 is a block diagram illustrating a typical prior art hierarchically-indexed data
19 store 200 and associated snapshots. The depicted hierarchically-indexed data store 200
20 includes a master data block 210, a master index block 220, one or more file index blocks
21 222, one or more data files 250, a snapshot tracking system 230 including a snapshot map
22 232, one or more snapshot files 240 and associated snapshot headers 234, snapshot indices
23 236, and archived data blocks 238.

24 The master data block 210 points to the master index block 220. The master index
25 block 220 points to file index blocks 222 which in the depicted example includes file index
26 blocks 222a through 222e. The depicted file index blocks 222a – 222c point to
27

1 corresponding data files 250a – 250c, creating a hierarchically-indexed system for each data
2 file location.

3 The data files 250 typically include a series of data blocks linked by pointers to
4 subsequent data blocks. For clarity, the data files 250a to 250c are shown as a single
5 contiguous segment. As the data files 250 are modified, blocks may be added to a particular
6 file by linking additional data blocks to the series of linked data blocks. Data blocks may also
7 be deleted from a file by eliminating links to the deleted blocks. Similarly, data blocks may
8 be modified within a file by overwriting the data in the linked blocks.

9 The file index blocks 222d and 222e point to snapshot files in the same manner as
10 file index blocks 222a – 222c point to data files. The depicted snapshot tracking system 230
11 is essentially a flat indexed file system that includes a map of snapshot instances 232 and
12 associated snapshot files. A snapshot instance 240a includes the snapshot header 234a, the
13 snapshot index 236a, and the archived data blocks 238a – 238c. The archived data blocks
14 238a – 238c are copies of data blocks from data files 250a – 250c before they were deleted or
15 overwritten as data files 250a – 250c are modified subsequent to a preceding snapshot 240b.
16 The snapshot 240b includes the snapshot header 234b, the snapshot index 236b and the
17 archived data blocks 238d – 238f.

18 Restoration of the snapshot instance 240a via a snapshot provider or the like allows
19 the hierarchically-indexed data store 200 to be restored to the exact state that existed when
20 snapshot instance 240a was taken. Successive application of snapshot instance 240a and
21 snapshot instance 240b allow the hierarchically-indexed data store 200 to be restored to the
22 exact state that existed when snapshot instance 240b was taken.

23 The current art, exemplified by the hierarchically-indexed data store 200 establishes
24 snapshot data structures that facilitate a complete and consistent image for recovery
25 purposes. Nevertheless, as depicted in Figure 2, the current art for hierarchically-indexed
26
27

1 systems is inefficient and complex compared to log-structured systems, and does not directly
2 support snapshot-based asynchronous replication to a target data store.

3 Consequently, a need exists for devices, methods and systems that perform
4 asynchronous log-like snapshot-based replication for non-log-structured file systems and
5 associated data stores. Such devices, methods, and systems would provide the advantages of
6 log-structured replication techniques to widely available hierarchically-indexed data storage
7 systems.

SUMMARY OF THE INVENTION

The various elements of the present invention have been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available data store replication means and methods. Accordingly, the present invention provides an improved apparatus, method, and system for asynchronous replication of a hierarchically-indexed data store.

In one aspect of the present invention, an apparatus for asynchronous file-based replication of a hierarchically-indexed data store includes a replication source having a hierarchically-indexed data store, a tracking module configured to track file regions that have changed on the replication source since a previous snapshot instance, and a replication module configured to communicate data contained within changed file regions in response to a subsequent snapshot instance.

The apparatus for asynchronous file-based replication of a hierarchically-indexed data store may also include a replication target configured to receive the data within changed file regions from the replication source, write the data within corresponding files regions on the replication target, and initiate a snapshot operation that synchronizes the replication target with the replication source.

In one embodiment, the replication module communicates the data contained within the changed file regions in an order that is independent of the order in which the data was written. Communicating in a different order facilitates optimized retrieval of the changed file regions from the hierarchically-indexed data store and any buffers or data caches associated therewith.

In certain embodiments, the apparatus includes a storage management module configured to set policies related to replication. The replication module may conduct replication operations as directed by such policies in order to reduce the complexity of managing replication operations.

1 The apparatus for asynchronous file-based replication of a hierarchically-indexed data
2 store may also include a snapshot module configured to provide snapshot services. In certain
3 embodiments, the snapshot module may be a standard snapshot service module provided by a
4 third party vendor or the like. In addition, the present invention uses a conventional file
5 system such as a hierarchical file system and need not use a log-structured file system for
6 efficient operation as required by previous solutions. The ability to use a standard snapshot
7 service module and a conventional file system increases the usefulness of the present
8 invention and facilitates deployment on a wide range of systems.

9 In certain embodiments, the tracking module uses a file system driver to track file
10 regions that have changed since a previous snapshot instance. In one embodiment, the file
11 system driver is an installable driver.

12 In another aspect of the present invention, a method for asynchronous file-based
13 replication of a hierarchically-indexed data store includes storing data on a hierarchically-
14 indexed data store, tracking file regions that have changed since a first snapshot instance, and
15 communicating data contained within changed file regions in response to a second snapshot
16 instance.

17 In certain embodiments, the data contained within the changed file regions may be
18 communicated in an order different from the order in which the data is written to the data
19 store. Data communication may also be conducted as directed by policies related to
20 replication. In one embodiment, the method repeats replication operations at a configurable
21 time interval.

22 Various elements of the present invention may be combined into a system for
23 asynchronous file-based replication of a hierarchically-indexed data store. In one
24 embodiment, the system includes a target volume, a source volume corresponding to a
25 hierarchically-indexed data store, a tracking module configured to track file regions that have
26
27

1 changed since a first snapshot instance, and a replication module configured to communicate
2 data contained within changed file regions in response to a second snapshot instance.

3 The various elements and aspects of the present invention facilitate conducting
4 replication operations on a hierarchically indexed data store with the same ease as is
5 customarily associated with log-structured data stores. The present invention reduces the
6 bandwidth and buffering requirements associated with synchronous replication and facilitates
7 asynchronous replication of data in a flexible manner. These and other features and
8 advantages of the present invention will become more fully apparent from the following
9 description and appended claims, or may be learned by the practice of the invention as set
10 forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the advantages and objects of the invention are obtained will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof, which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

Figure 1 is a block diagram illustrating a typical prior art log-structured mirrored data store and associated snapshots;

Figure 2 is a block diagram illustrating a typical prior art hierarchically-indexed data store and associated snapshots;

Figure 3 is a block diagram illustrating one embodiment of an asynchronous replication system of the present invention;

Figure 4 is a flow chart illustrating one embodiment of a changed regions tracking method of the present invention;

Figure 5 is a flow chart illustrating one embodiment of a target update method of the present invention; and

Figure 6 is text-based diagram illustrating a change tracking data structure in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, modules may be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions that may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module. For example, a module of executable code could be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices.

Modules may also be implemented in hardware as electronic circuits comprising custom VLSI circuitry, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Figure 3 is a block diagram illustrating one embodiment of an asynchronous replication system 300 of the present invention. The depicted embodiment of the asynchronous replication system 300 includes a replication source 310, a communication link 320, and a replication target 330. The communication link 320 supports command and data transfer between a data replication module 318 associated with a source data store 304a and

1 an update module 336 associated with a target data store 304b. In one embodiment, the
2 source data store 304a and the target data store 304b may be the same data store or may be
3 co-located within the same storage subsystem. In another embodiment, the target data store
4 304b is remotely located from the source data store 304a.

5 The replication source 310 and the replication target 330 include a number of
6 elements related to data replication, including one or more data stores 304, a file system 312,
7 and a snapshot service 316. The depicted embodiment of the data source 330 also includes
8 an application program 302, a snapshot management module 304, a tracking module 314,
9 and a replication module 318, while the depicted embodiment of the replication target 330
10 further includes an update module 336. A tracking mechanism, such as a block map 315 may
11 be used and is in one embodiment an example of a data structure used for tracking changes
12 during snapshot operations.

13 As depicted, the data stores 304 may contain one or more volumes comprised
14 of logical subunits such as blocks, a volume may contain one or more files, and a file may
15 contain one or more regions such as blocks or pages. A volume block map (not shown) may
16 also be associated with each data store 304. In one embodiment, the volume block map is a
17 data structure that includes an entry for the overwrite status of all block addresses in the
18 volume.

19 The application program 302 uses the file system 312 to conduct file-oriented
20 operations such as writing data to and reading data from the data store 304. The snapshot
21 service 316 in one embodiment manages snapshot operations on the data store 304, as
22 discussed for the prior art hierarchically-indexed data store 200 of Figure 2. While the
23 specific instance of a snapshot is discussed herein by way of example, one skilled in the art
24 will readily recognize that the present invention may be employed using any type of point-
25 in-time image replication operation.

26 The tracking module 314 keeps track of those regions that have changed within each
27 file on the source data store 304a between successive snapshots (*i.e.* over a particular

1 snapshot interval). In certain embodiments, as each snapshot is initiated, the tracking module
2 314 clears one or more block maps 315 that track changes to files on the data store 304 from
3 memory. One example of a suitable block map 315 is discussed below with respect to Figure
4 6. In certain embodiments, the tracking module 314 clears one or more block maps under
5 direction of the replication module 318 as directed by snapshot management policies.

6 The replication module 318 interacts with the file system 312 to read the data blocks,
7 or the like, from the source data store 304a, that have changed between successive snapshot
8 instances. The retrieved data blocks may be transmitted to the replication target. If multiple
9 changes were made to a particular data block during a snapshot interval, the last version of
10 the data block is retrieved from the source data 304a and communicated to the replication
11 target 330. By retrieving and communicating the last version corresponding to a snapshot
12 interval, the processing and communication burden associated with replication may be
13 significantly reduced.

14 In one embodiment, the replication module 318 uses one or more block maps 315 to
15 ascertain the file regions that have changed over a particular snapshot interval. One example
16 of a suitable block map 315 is the data structure 600 of Figure 6. In the aforementioned
17 embodiment, the tracking module 314 and the replication module 318 rotate between two or
18 more block maps residing in memory. The replication module 318 may also interact with the
19 update module 338 to transfer the changed data blocks to the replication target 330 over the
20 communication link 320.

21 In response to the reception of the changed data blocks at the replication target 330,
22 the update module 338 ensures that the changed data blocks are written to a proper location
23 on the target data store 304b. In conjunction with writing the changed data blocks to the
24 target data store 304b, the update module 338 may also interact with the snapshot service
25 316b to synchronize a snapshot image located on the source data store 304a with a
26 corresponding snapshot image located on the target data store 304b. In one embodiment,
27 synchronization of the snapshot images is accomplished by invoking a snapshot operation on

1 the target data store 304b after all of the changed data blocks have been written to the target
2 data store 304b.

3 The storage management module 306 may be included in certain embodiments of the
4 present invention. The depicted storage management module 306 provides an interface and
5 associated mechanisms for managing policies related to storage. For example, policies may
6 establish the criteria for conducting snapshot or other data replication operations. In one
7 embodiment, the criteria include the elapsed interval since a previous snapshot, the amount
8 of available storage, the amount of recently changed data, and the like.

9 To implement policies, the storage management module 306 may communicate with
10 the tracking module 314, the snapshot service 316a, and the file system 312a. The
11 relationship between the storage management module and the aforementioned modules need
12 not be exclusive. For example, the services provided by the snapshot service 316a, such as
13 snapshot operations, may be invoked by an application, a utility, or the like, without affecting
14 snapshot-based replication operations.

15 Figure 4 is a flow chart illustrating one embodiment of a changed regions tracking
16 method 400 of the present invention. The depicted embodiment of the changed regions
17 tracking method 400 includes an initialize step 410, a store data step 420, a track changed
18 regions step 430, a snapshot request test 440, a store tracking data step 450, and a
19 communicate changed regions step 460. In one embodiment, the method 400 may be
20 conducted using the system 300 of Figure 3.

21 As depicted, the changed regions tracking method 400 starts coincident with creating
22 a snapshot instance. The initialize step 410 initializes any data structures involved with
23 tracking changed regions on a storage volume or portion thereof, such as a file. In one
24 embodiment, the initialize step 410 constructs a blank block map for each volume to be
25 replicated.

26 Subsequent to the initialize step 410, the depicted method 400 proceeds to the store
27 data step 420. The store data step 420 stores data on a data store such as the source data store

1 304a depicted in Figure 1. The store data step 420 may be conducted in response to a write
2 request by the application program 302, or the like.

3 The track changed regions step 430 updates one or more data structures related to
4 tracking changes. In one embodiment, the track changed regions step 430 sets one or more
5 status fields within a block map 315. The volume block map 315 may be implemented with
6 a suitable data structure such as that shown in Figure 6. Preferably, the status fields indicate
7 the updated status of all blocks written during the store data step 420. In one embodiment,
8 the changed regions are file regions that were changed between snapshot instances. In
9 another embodiment, knowledge of the relationship between the block-level changes and file
10 region changes (as stored within a snapshot) facilitates using block-level changes in place of
11 file-level changes in the present invention.

12 As depicted in Figure 4, the store data step 420 and track changed regions step 430
13 iterate until the snapshot request test 440 is affirmative. In response to an affirmative
14 snapshot request, the method proceeds to the store tracking data step 450. Otherwise, the
15 changed regions tracking method 400 loops to the store data step 420.

16 At the store tracking data step 450, the tracking data is captured and stored for use
17 with the communicate changed regions step 460. In one embodiment, the store tracking data
18 step 450 involves storing a block map 315 (e.g., the data structure 600 of Figure 6)
19 containing status information indicating whether a block associated with a file region has
20 been changed. In certain embodiments, multiple tracking data structures may be rotated
21 between the tracking module 314, and the replication module 318, and the store tracking data
22 step 450 may be omitted.

23 The communicate changed regions step 460 communicates to a replication target the
24 file regions that have changed over a snapshot interval. The communicate changed regions
25 step 460 may be conducted asynchronous to the rest of the method 400. One advantage of
26 asynchronous execution is that the communicate changed regions step 460 may be deferred
27 to a convenient time or it may be conducted as a background process.

1 The communicate changed regions step 460 may communicate the changes to a
2 replication target in any suitable order. For example, the changes may be sorted in an order
3 that optimizes retrieval of the changed data from a data store. In certain embodiments, the
4 communicate changed regions step 460 extracts information from the tracking data structures
5 stored during the store tracking data step 450 and includes that information in the conducted
6 communications.

7 Subsequent to completion of the communicate changed regions step 460, the depicted
8 changed regions tracking method 400 terminates at an end step 470. The method 400
9 facilitates snapshot-based data replication from a replication source having a hierarchically-
10 indexed data store, or the like. While the depicted method 400 is shown in a certain
11 sequential order, many real-time, multi-threaded, or event driven forms of the present method
12 may be derived under the present invention in a manner that will be readily apparent to those
13 of skill in the art.

14 Figure 5 is a flow chart illustrating one embodiment of a target update method 500 of
15 the present invention. The depicted embodiment of the target update method 500 includes a
16 receive changed data step 510, a write changed regions step 520, a conduct snapshot step
17 530, and an end step 540. The target update method 500 may be conducted in conjunction
18 with a replication target such as the replication target 330 depicted in Figure 3.

19 The receive changed data step 510 receives the changed regions associated with a
20 snapshot interval on a source system. In one embodiment, the regions are file regions that
21 were changed between snapshot instances. In another embodiment, the regions are block-
22 level changes to a volume or portion thereof. Subsequent to the receive changed data step
23 510, the target update method 500 proceeds to the write changed regions step 520.

24 The write changed regions step 520 writes data received from the replication source
25 to a target volume such as the replication target 330. In one embodiment, the replication
26 source and target are essentially identical (including identical file systems, snapshot services,
27 and snapshot storage formats), and the data is written as block level changes. In another

1 embodiment, the file system on the replication target is different than the replication source
2 and the changes are written as file-level changes.

3 The conduct snapshot step 530 synchronizes the replication target with the replication
4 source by conducting a snapshot operation subsequent to writing all the changed regions
5 associated with a snapshot interval on the replication source. Synchronizing the replication
6 target with the replication source facilitates conducting recovery operations in the event of a
7 disaster, or the like, or conducting rollback operations, such as restarting payroll processing.

8 Some aspects of the target update method 500 may be conducted concurrently. For
9 example, in one embodiment, the write changed regions step and the conduct snapshot step
10 may be conducted concurrently with the receive changed data step 510 associated with a
11 subsequent snapshot. In certain embodiments, additional replication-related write operations
12 (i.e. the write changed regions step 520) associated with subsequent snapshots are blocked
13 until a presently conducted snapshot operation is completely finished.

14 Figure 6 is a text-based diagram illustrating one embodiment of a change tracking
15 data structure 600 of the present invention. The change tracking data structure 600, or a data
16 structure with a similar function, may be used by the tracking module 314, the replication
17 module 318, the update module 338, the changed regions tracking method 400, and the target
18 update method 500 in order to conduct snapshot-based replication operations between a
19 replication source, such as the replication source 310, and a replication target, such as the
20 replication target 330.

21 As depicted, the change tracking data structure 600 includes a block index 610, a
22 block address 620, a status indicator 630, a file index 640, and a region address 650. The
23 depicted embodiment of the change tracking data structure 600 is essentially a block map
24 corresponding to a specific snapshot interval. The change tracking data structure 600 may
25 also include other information useful in tracking changes. Such information may include
26 caching information.

1 The block index 610 functions essentially as a logical identifier for a block associated
2 with a particular volume, or the like. In contrast, the block address 620 preferably identifies
3 a physical location of the actual block within the physical media of a storage device. In one
4 embodiment, volumes may be distributed across multiple storage devices and the block
5 address also includes a device identifier.

6 The status indicator 630 indicates the status of the block, particularly whether a
7 change was made to the block since a previous snapshot. The file index 640 identifies the
8 file associated with the data block, while the region address 650 indicates the file region
9 associated with the particular data block.

10 As implied by the flexibility of the change tracking data structure 600, the present
11 invention may be used to replicate specific files or entire volumes to a replication target.
12 Consequently, the present invention may be used to maintain point-in-time retrievable
13 images of a volume or portion thereof on a target volume while reducing the amount of data
14 necessary to replicate and store source volume changes on the target volume. In addition, the
15 present invention works with conventional hierarchically-indexed file systems and data stores
16 and does not require a log-structured file system and associated data store for efficient
17 operation.

18 The present invention may be embodied in other specific forms without departing
19 from its spirit or essential characteristics. The described embodiments are to be considered
20 in all respects only as illustrative and not restrictive. The scope of the invention is, therefore,
21 indicated by the appended claims rather than by the foregoing description. All changes
22 which come within the meaning and range of equivalency of the claims are to be embraced
23 within their scope.

24 What is claimed is:
25
26
27